Jet finding Algorithms at Tevatron

B.Andrieu (LPNHE, Paris)

On behalf of the DØ collaboration

Outline:

- Introduction
- The Ideal Jet Algorithm
- Cone Jet Algorithms: RunII/RunI, D0/CDF
- $k_\perp$ Jet Algorithm
- Summary

Jets: from parton to detector level

$$\sigma^{p\bar{p}\to jets} = \int d\Omega \sum_{ij} f_{i/\bar{p}}(x_{\bar{p}},\mu_{\bar{p}}^2)\, f_{j/p}(x_p,\mu_p^2)\, d\sigma^{ij\to kl}$$

$d\sigma^{ij\to kl}$ partonic cross section $\propto \alpha_S^2(\mu_R^2)$

$f_{i/\bar{p}}(x_{\bar{p}},\mu_{\bar{p}}^2)\,(f_{j/p}(x_p,\mu_p^2))$ PDF of parton i (j) in p (p)

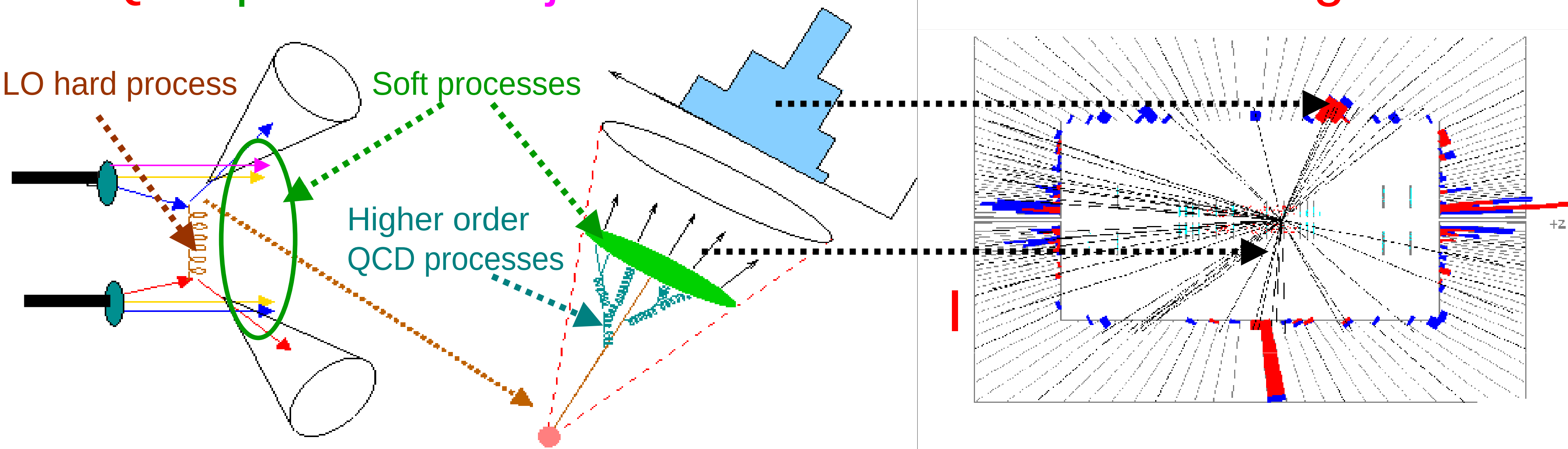
μ_R renormalisation scale

$\mu_{\bar{p}}(\mu_p)$ factorisation scales in p (p)

QCD $\Rightarrow$ quarks and gluons at high p_T produce jets (Sterman&Weinberg, 1977)

Non-perturbative processes not predictable → QCD inspired phenomenology

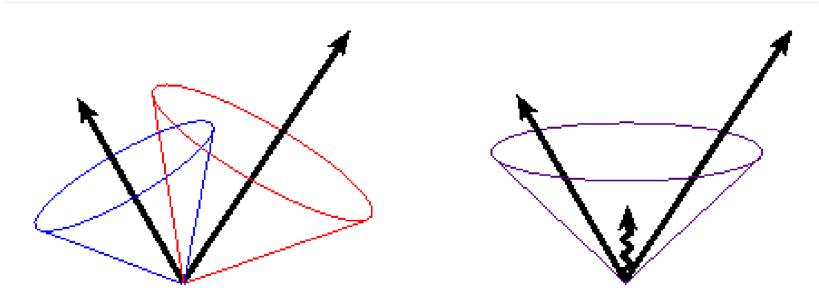
QCD partons → jets of hadrons → detector signals



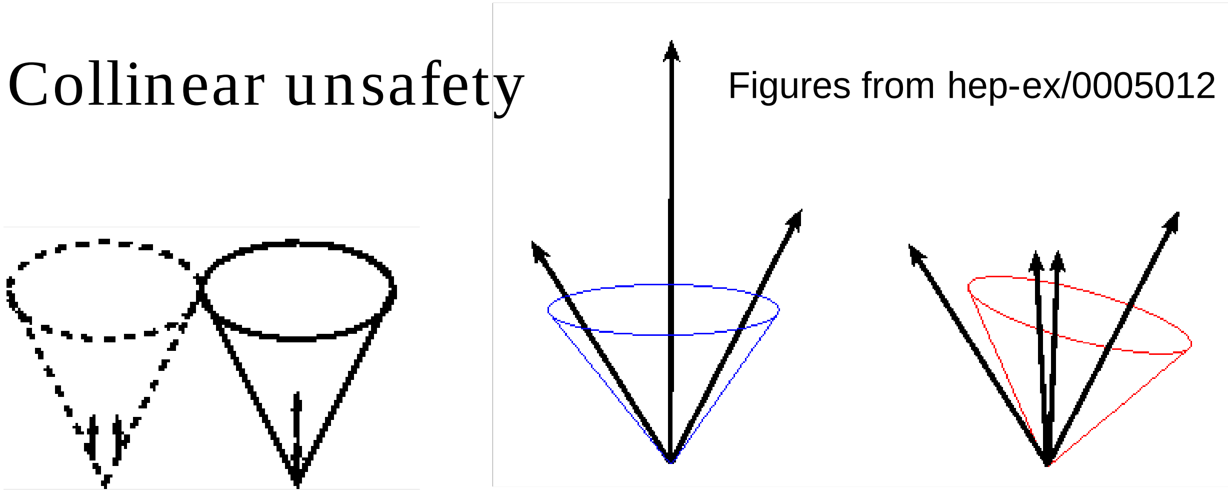
Jets: from parton to detector level

QCD $\Rightarrow$ { Quark and gluon jets (identified to partons) can be compared to detector jets, if jet algorithms respect collinear and infrared safety (Sterman&Weinberg, 1977)

Infrared unsafety



Collinear unsafety



Figures from hep-ex/0005012

High E_T jets $\Leftrightarrow$ "Hard" QCD

(non-perturbative effects & scale uncertainty reduced)

- ⇨ Direct insight into parton dynamics
- ⇨ Precise tests of perturbative QCD predictions
- ⇨ Measure α_S, constrain proton PDFs, ...
- ⇨ Search for new physics

Low E_T jets $\Leftrightarrow$ "Soft" QCD

(non-perturbative effects & scale uncertainty important)

- ⇨ Test phenomenological models (underlying event, fragmentation)
- ⇨ Study detailed jet structure (jet shapes)

Jet definition

Two things need to be done to define a jet:

- Associate "close" to each other "particles"
  → **Clustering (Jet Algorithm)**
  - "particles" can be:
    - partons (analytical calculations or parton showers MC)
    - "hadrons" = final state particles (MC particles or charged particles in trackers)
    - towers (or cells or preclusters or any local energy deposits)
  - "close" ? → Distance
    - independent of the distance from interaction point
    - invariant under longitudinal boosts

  → $\Delta R = \sqrt{\Delta\eta^2+\Delta\phi^2}$ or $\sqrt{\Delta Y^2+\Delta\phi^2}$ (preferred in RunII) for Cone Algorithm
  → relative p_T for $k_\perp$ algorithm

- Calculate jet 4-momentum from "particles" 4-momenta
  → **Recombination scheme**
  - invariant under longitudinal boosts
    → Snowmass scheme (RunI): E_T-weighted recombination scheme in (η,ϕ)
    → covariant or E-scheme (preferred for RunII): 4-momenta addition
  - used at the end of clustering but also during clustering process
    (not necessarily the same, still preferrable)

The ideal jet algorithm for $p\bar{p}$

Compare jets at the parton, hadron and detector level

⇨ **Jet algorithms should ensure**

General

- infrared and collinear safety
- invariance under longitudinal boosts
- fully specified and straightforward to implement
- same algorithm at the parton, hadron and detector level

Theory

- boundary stability (kinematic limit of inclusive jet cross section at $E_T = \sqrt{s}/2$)
- factorisation (universal parton densities)

Experiment

- independence of detector detailed geometry and granularity
- minimal sensitivity to non-perturbative processes and multiple scatterings at high luminosity
- minimization of resolution smearing/angle bias
- reliable calibration
- maximal reconstruction efficiency (find all jets) vs minimal CPU time
- replicate RunI cross sections while avoiding theoretical problems

Run I Cone Algorithm

- Based on Snowmass algorithm: E_T-weighted recombination scheme in (η,ϕ)
- **Preclustering** (D0, similar algorithm for CDF)
  Note: Tower segmentation in (η,ϕ) space: **D0 → 0.1 X 0.1**, **CDF → 0.11 X 0.26**
  - start from seeds (= hadronic towers with p_T >1 GeV ordered in decreasing p_T)
  - cluster (and remove) all contiguous calorimeter towers around seed in a R= 0.3 cone
- **Clustering**
  - start from preclusters (ordered in decreasing E_T)
  - proto-jet candidate = all particles within R_{cone} of the precluster axis in (η,ϕ) space
    CDF: keep towers of the original precluster through all iterations (ratcheting)
  - proto-jet direction compared before/after recombination → iterate until it is stable
- **Merging/Splitting** (treat overlapping proto-jets)
  - $E_{1\cap 2} > f \,.\, \mathrm{Min}(E_1,E_2)$ → Merge jets
  - $E_{1\cap 2} < f \,.\, \mathrm{Min}(E_1,E_2)$ → Split jets = assign each particle to its closest jet
  - **D0: f = 50 %, use only clusters with E_T > 8 GeV** - **CDF: f = 75 %**
- **Final calculation of jet variables** (modified Snowmass scheme)
  - scalar addition of E_T (D0) or E (CDF) of particles to determine jet E_T or E
  - addition of 3-momenta of particles to determine jet direction, then (η,ϕ)
    Note: this procedure is not Lorentz invariant for boosts along beam axis
    CDF: $E_T = E \sin(\theta)$

Why new algorithms for Run II?

Run I Cone algorithms have many drawbacks

- Different in D0 and CDF
- Not infrared and collinear safe due to the use of seeds
  (collinear safety ensured at sufficiently large E_T : E_T >20 GeV with p_T^{min} (seed) = 1 GeV in D0)
- Preclustering difficult to match at parton or hadron level
- CDF ratcheting not modelled in theory
- Need to introduce a new parameter (R_{sep}) in jet algorithm at parton level to match theory predictions to measurements
  (S.D. Ellis et al., PRL69, 3615 (1992))
- Not invariant under boosts along beam axis

→ 2 new Cone Algorithms proposed for RunII
(G.C. Blazey et al., "RunII Jet Physics", hep-ex/0005012)

- Seedless Cone Algorithm
- RunII (= Improved Legacy or Midpoint) Cone Algorithm

→ Use $k_\perp$ algorithm (already used in RunI)

Seedless Cone Algorithm

Not really "seedless"

→ Use enough seeds (all towers) to find all stable cones

- First step:
  - form cone around seed, recalculate cone direction (Snowmass recombination)
  - stop processing seed if the cone centroid is outside of the seed tower
    CDF: use tower size X 1.1 to avoid boundary problems
- Secund step similar to Run I Cone algorithm:
  - use the cones formed in first step (pre-protojets) as seeds
  - form cone around seed and recalculate cone direction (E-scheme = 4-momentum addition)
  - iterate until cone direction after/before recombination is stable
- Streamlined (faster) option
  - Stop iteration in second step if the cone centroid is outside of the seed tower
    → Only miss low E_T protojets or stable directions within the same tower

→ Infrared and collinear safe

→ Probably close to Ideal for a Cone algorithm

→ Even the streamlined version is very computational intensive

⇒ Use an approximation of Seedless Algorithm → RunII Cone

RunII Cone Algorithm (hep-ex/0005012)

How to build a valid approximation of the seedless algorithm?

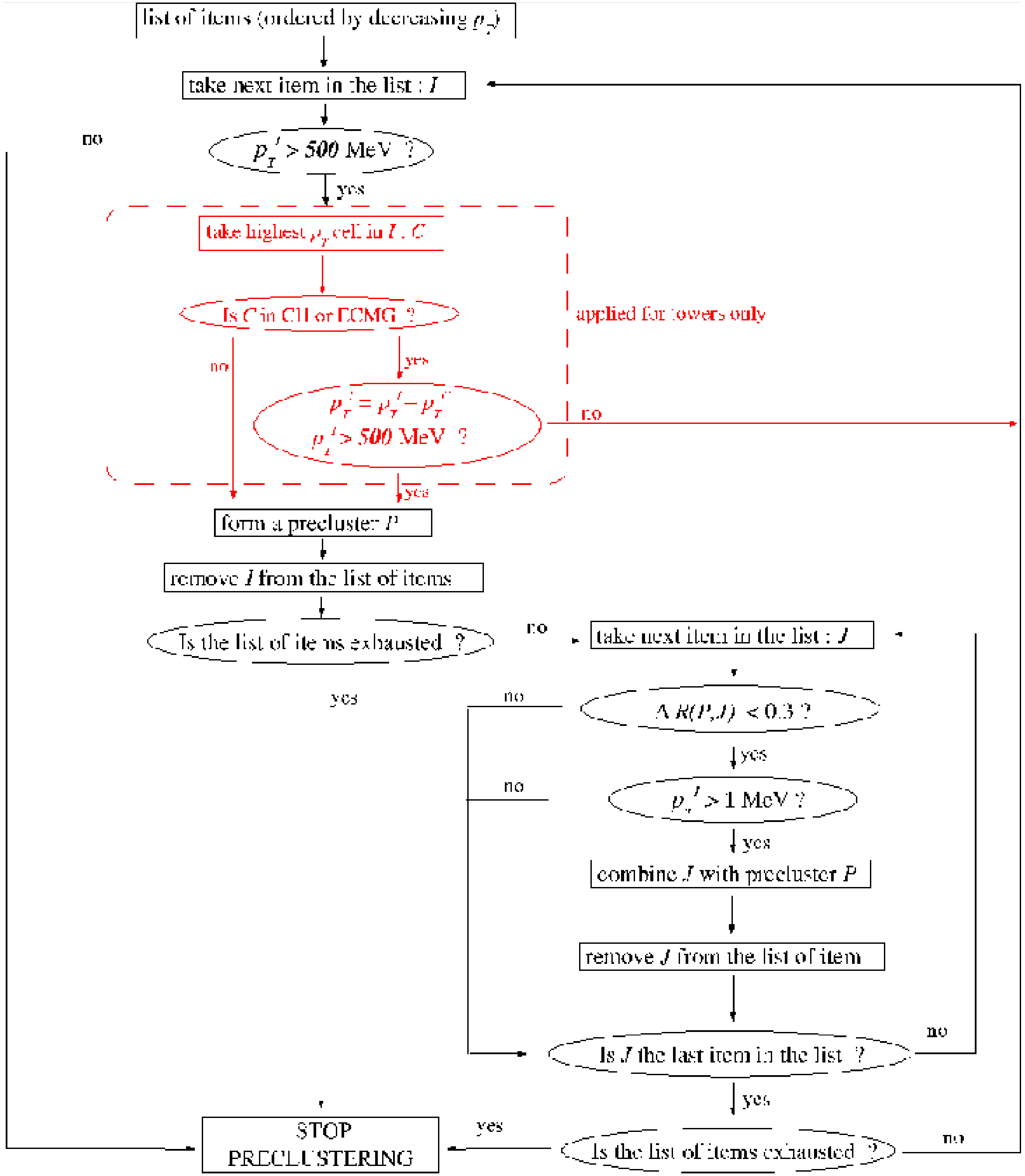
- QCD calculation at fixed order N
  → only $2^N - 1$ possible positions for stable cones (p_i , $p_i + p_j$, $p_i + p_j + p_k$,...)
- Data: consider seeds used in RunI Cone algorithms as partons
  → in addition to seeds, use 'midpoints' i.e. $p_i + p_j$, $p_i + p_j + p_k$,...
- only need to consider seeds all within a distance $\Delta R < 2R_{cone}$
- only use midpoints between proto-jets (reduce computing time)
- otherwise algorithm similar to RunI

Other specifications of the suggested RunII cone Algorithm

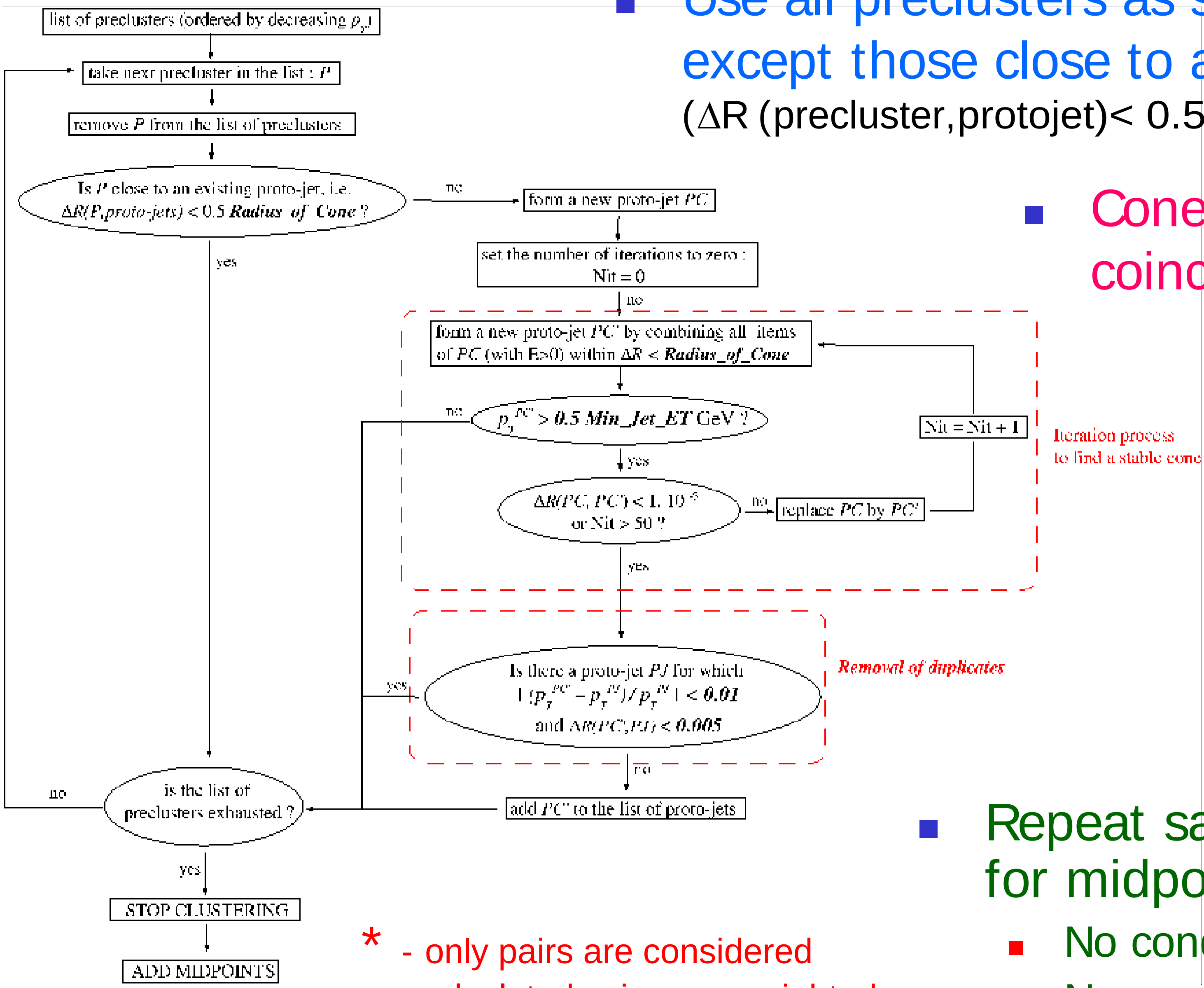
- E-scheme recombination = 4-momenta addition
- use true rapidity Y instead of pseudo-rapidity η in ΔR
- use all towers as seeds ($p_T > 1$ GeV)
- splitting/merging: p_T ordered, f = 50 %

D0 Run II Cone Algorithm: Preclustering

- Simple Cone Algorithm
- Start from particles with highest p_T and $p_T > 500$ MeV
- Precluster formed from all particles within a cone of r = 0.3 (r = 0.2) for Cone jets with R $\geq$ 0.5 (R = 0.3)
  (≠RunI: only neighbouring cells)
- Remove particles as soon as they belong to a precluster
- No cone drifting
- Precluster 4-momentum calculated using the E-scheme



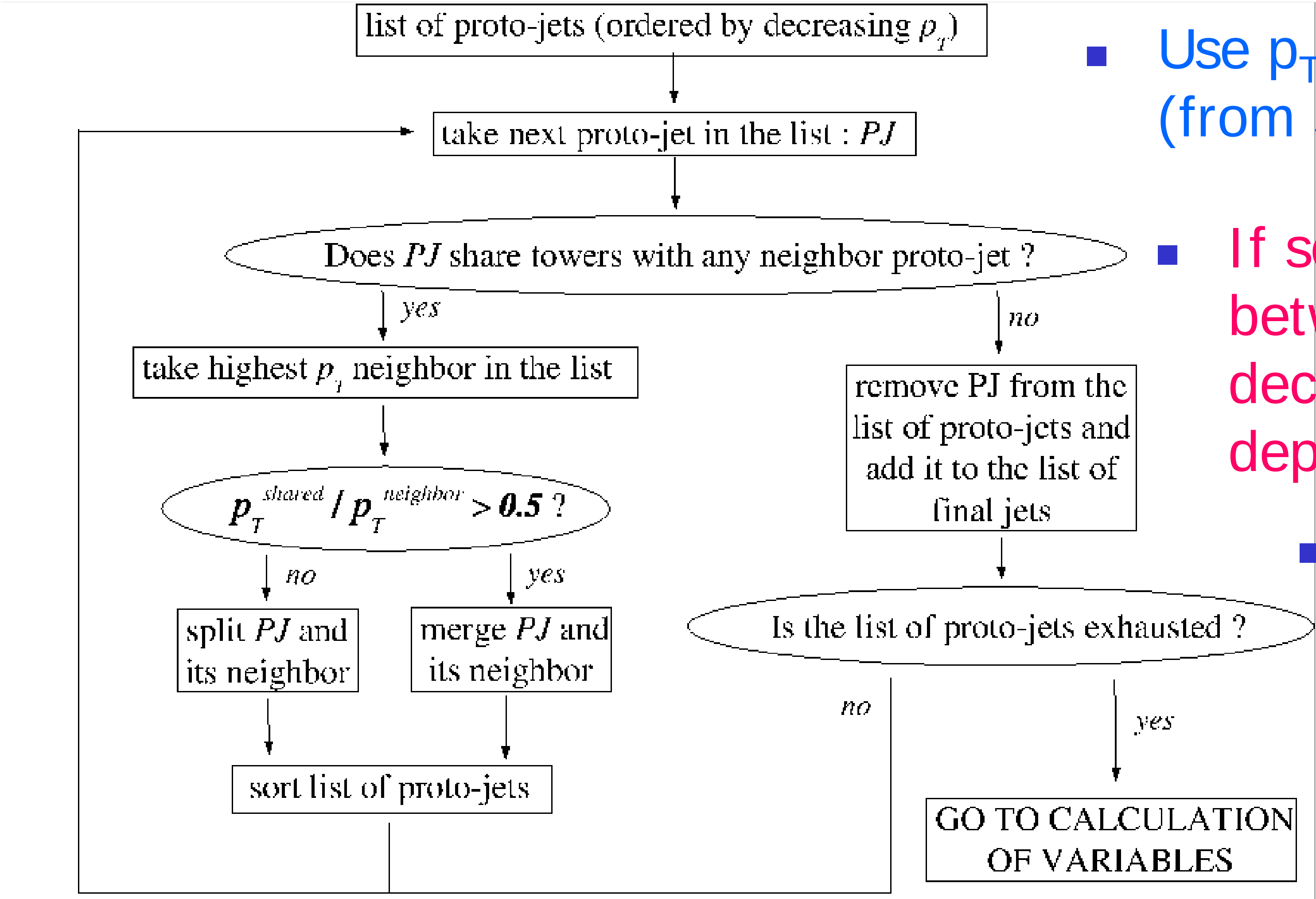
D0 Run II Cone Algorithm: Clustering



- Use all preclusters as seeds (p_T ordered), except those close to already found protojets
  (ΔR (precluster,protojet)< 0.5 R_{cone})
- Cone drifting until cone axis coincides with jet direction
- Abort drifting if:
  - p_T < 0.5 Jet p_T^{min}
  - # Iterations = 50
    (avoids infinite cycles)
- Remove duplicates
- Repeat same clustering for midpoints* except:
  - No condition on close protojet
  - No removal of duplicates

* - only pairs are considered
- calculated using p_T–weighted mean

D0 Run II Cone Algorithm: Merge/Split



- Use p_T ordered list of proto-jets (from seeds and midpoints)
  - If some energy is shared between two proto-jets, decide to split/merge depending on shared fraction
    - Recalculate 4-momenta of merged/splitted jets
      - Re-order list of merged/splitted jets

The Smaller Search Cone Algorithm

- Jets might be missed by RunII Cone Algorithm (S.D. Ellis et al., hep-ph/0111434)
  → low p_T jets
  - too close to high p_T jet to form a stable cone (cone will drift towards high p_T jet)
  - too far away from high p_T jet to be part of the high p_T jet stable cone
- proposed solution
  - remove stability requirement of cone
  - run cone algorithm with smaller cone radius to limit cone drifting
    ($R_{search} = R_{cone} / \sqrt{2}$)
  - form cone jets of radius R_{cone} around protojets found with radius R_{search}

Remarks

- Problem of lost jets seen by CDF, not seen by D0
  → A physics or an experimental problem?
- Proposed solution not satisfactory in terms of elegance and simplicity

⇒ D0 prefers using RunII Cone without Smaller Search Cone

$k_\perp$ Algorithm

Description of inclusive $k\perp$ algorithm (Ellis&Soper, PRD48, 3160, (1993))

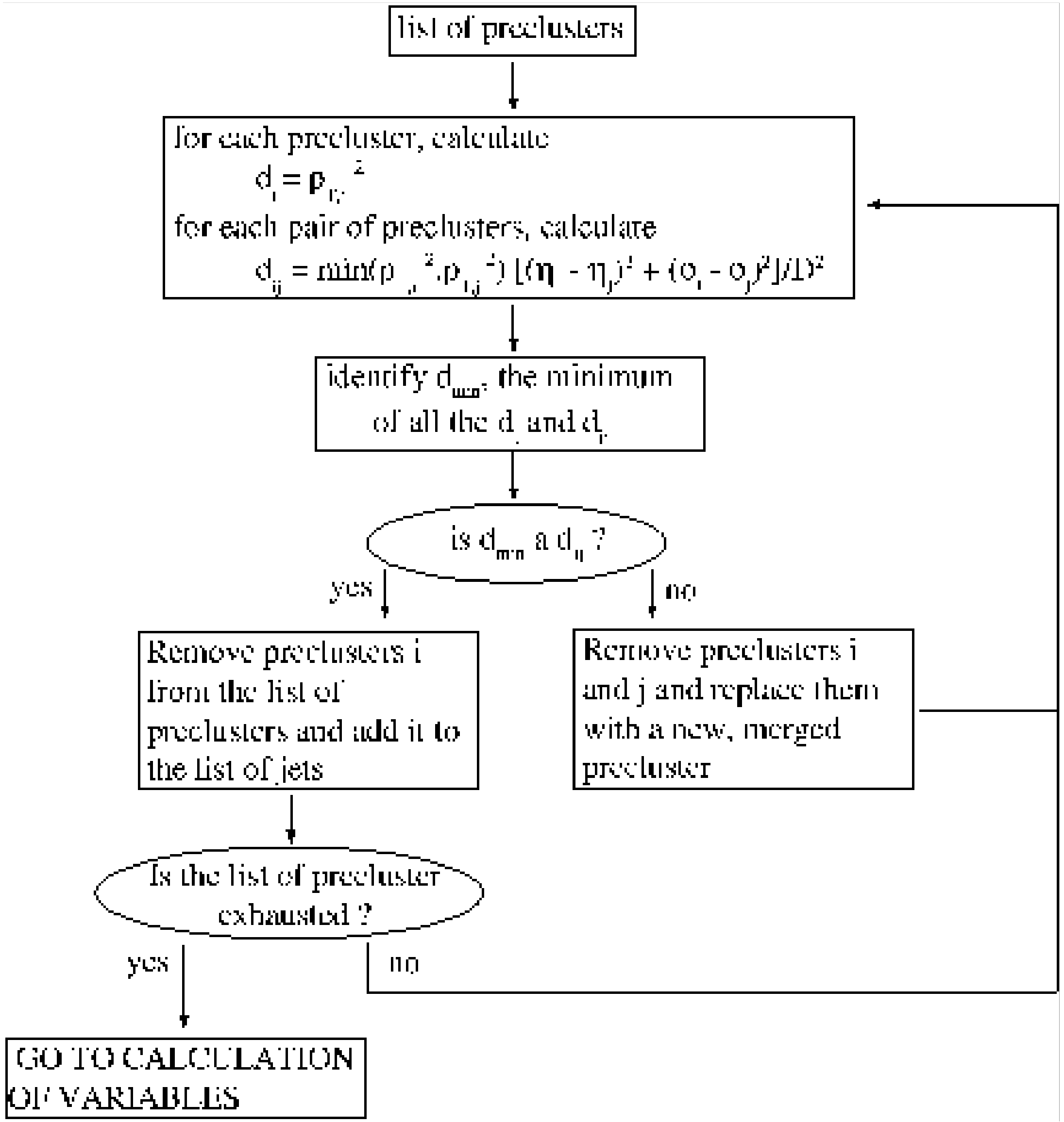
- p_T ordered list of particles → form the list of $d_i = (p_T^i)^2$
- calculate for all pairs of particles, $d_{i\,j} = \mathrm{Min}((p_T^i)^2, (p_T^j)^2)\ \Delta R/D$
- find the minimum of all d_i and $d_{i\,j}$
  - if it is a d_i , form a jet candidate with particle i and remove i from the list
  - if not, combine i and j according to the E-scheme
  - use combined particle i + j as a new particle in next iteration
  - need to reorder list at each iteration → computing time $\propto O(N^3)$ (N particles)
- proceed until the list of preclusters is exhausted

Remarks

- originally proposed for e^+e^- colliders, then adapted to hadron colliders (S. Catani et al., NPB406,187 (1993))
- universal factorisation of initial-state collinear singularities
- infrared safe: soft partons are combined first with harder partons → result stable when energy of soft partons -> 0
- collinear safe: two collinear partons are combined first in the original parton
- no issue with merging/splitting

D0 Run II $k_\perp$ Algorithm

- Use E -scheme for recombination
- Use p_T ordered list of preclusters (geometrical 2x2 preclustering)
- Remove preclusters with $E < 0$
- Either merge pairs of preclusters which are closest to each other in relative p_T or form a jet with each isolated low p_T precluster
- When all preclusters have been associated to a jet, calculate 4-momenta of all jets
- Apply a p_T^{min} cut on jets ($p_T > 8$ GeV)



Summary

- RunII (Midpoint) Cone Algorithm clear improvement over RunI Algorithm
- Many problems or questions still remain open (not exhaustive list):
  - D0 uses only RunII Cone (Midpoint) Algorithm (no smaller search cone)
  - CDF still uses JetClu (RunI) Cone Algorithm + Smaller Search Cone Algorithm
  - D0 implementation does not fully follow RunII Cone recommendations
    - $p_T^{min} / 2$ cut on proto-jets candidates
    - preclustering
    - seeds too close to already found protojets not used
  - influence of parameters for precluster formation?
  - usefulness of a p_T cut on proto-jets before merging/splitting at high luminosity?
  - procedure chosen for merging/splitting optimal?
  - origin of the difference D0 vs CDF for lost jets problem?
- In contrast, $k_\perp$ algorithm is conceptually simpler, theoretically well-behaved, although less intuitive. It also needs studies, as for the RunII Cone Algorithm (jet masses, sensitivity to experimental effects, ...).

$\Rightarrow$ However, shouldn't we put more effort on using $k_\perp$ algorithm and less on reproducing results obtained with RunI algorithms? (personal statement)